Improving parallel scalability for edge plasma transport simulations with neutral gas species

M. McCourt,[1,3] T. D. Rognlien,[2] L. C. McInnes,[3] and H. Zhang[3]

[1] Center for Applied Mathematics
657 Rhodes Hall
Cornell University, Ithaca, NY 14853

[2] Fusion Energy Sciences Program
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550

[3] Argonne National Laboratory
Mathematics and Computer Science Division
9700 South Cass Avenue, Argonne, IL 60439

E-mail: mccomic@mcs.anl.gov

Abstract. Simulating the transport of multi-species plasma and neutral species in the edge region of a tokamak magnetic fusion energy device is computationally intensive and difficult due to coupling among various components, strong nonlinearities, and a broad range of temporal scales. In addition to providing boundary conditions for the core plasma, such models aid in the understanding and control of the associated plasma/material-wall interactions, a topic that is essential for the development of a viable fusion power plant. The governing partial differential equations are discretized to form a large nonlinear system that typically must be evolved in time to obtain steady-state solutions. Fully implicit techniques using preconditioned Jacobian-free Newton-Krylov methods with parallel domain-based preconditioners are shown to be robust and efficient for the plasma components. Inclusion of neutral gas components, however, increases the condition number of the system to the point where improved parallel preconditioning is needed. Standard algebraic preconditioners that provide sufficient coupling throughout the global domain to handle the neutrals are not generally scalable. We present a new preconditioner, termed FieldSplit, which exploits the character of the neutral equations to improve the scalability of the combined plasma/neutral system.

AMS classification scheme numbers: 65F08

PACS numbers: 52.65.Kj

1. Introduction

Many systems of partial differential equations arising in engineering and physics applications exhibit a wide range of temporal and spatial scales; increasingly, these already complicated models are being coupled together for multi-physics simulations [1]. Examples of this multi-physics approach in numerical plasma simulation for magnetic fusion devices can be found in [2, 3, 4]. Applications from other areas include porous media [5] and fission power plants [6]. In each of these examples, knowledge about the characteristics of components of the system is used to produce a stable and efficient simulation that may provide new insights into the physics or mathematical aspects of the models.

This paper focuses on a numerical model that describes the transport of plasma and neutral species in the edge region of tokamak magnetic fusion energy devices in toroidal geometry. Transport simulations discussed here are conducted by using UEDGE [7, 8]. The strong external toroidal magnetic field ($\mathbf{B}$) in these devices results in a large anisotropy for the plasma, with transport along the magnetic field (termed parallel) having a much faster timescale than transport across the magnetic field (termed perpendicular or radial). In this paper, "parallel" is used as an adjective in two different senses: one is transport along $\mathbf{B}$ just mentioned, and the second is using multiprocessor parallel algorithms; which meaning applies should be clear from the local context.

An accurate model of the edge plasma region in fusion devices is needed to answer key design and operational issues. Among these issues are the peak plasma heat-flux exhaust and associated erosion of material surfaces, the migration of impurities from

such material interactions to the hot core, the build-up of tritium fuel in walls, and the pumping of helium ash produced by core fusion reactions. To this end, UEDGE provides edge modeling capabilities in the FACETS project [9, 10, 2], which has a goal of coupled core-edge-wall modeling of ever-increasing fidelity for magnetic fusion energy devices.

Modeling such plasmas in the core and the edge regions results in ill-conditioned problems due to the large range of time scales, the complicated geometry with a range of spatial scales, and the competing physical effects of the various degrees of freedom. In the edge region, the importance of the neutral gas component further complicates the problem. This paper focuses on the parallel scalability of the edge system, especially on the impact of the neutrals. Using early computational results and plasma/neutral transport knowledge, we construct a "physics-based" preconditioner that improves solution time and scalability over simpler algebraic and domain decomposition preconditioners. The term "physics-based" [11] can be ambiguous; we use it here to describe an operator that uses more than just algebraic insight to improve the conditioning of a linear system. A challenge in physics-based preconditioning is determining how to incorporate key physical characteristics into the operator. When this problem is handled successfully, the numerical performance can benefit greatly, as demonstrated for plasma transport in [12, 13, 14, 15].

We describe a series of computational experiments with the components of a coupled plasma/neutral simulation that illustrate why adding the neutral terms significantly increases the difficulty of the resulting linear system. The results of these initial simulations motivate a split approach to preconditioning, where groups of variables with similar physical attributes are solved with different methods tailored to the physics of each group. This approach differs from the approach of [14], where a semi-implicit approximation is made for the fully implicit system, which produces a preconditioner for the fully coupled system targeted to handle the different time scales with an easily invertible operator.

The paper is organized as follows: Section 2 introduces the relevant equations, the geometry, and the basic approach for the nonlinear solver using Jacobian-free Newton-Krylov methods. In Section 3 we consider plasma simulations with fixed neutral densities, and then complicate the simulation by allowing the neutrals to evolve. In Section 4 we analyze the conditioning of the neutral problem and develop a componentwise preconditioner (referred to as FieldSplit), which overcomes difficulties of other approaches; we also test the FieldSplit preconditioner on a variety of more complicated simulations. Section 5 discusses conclusions and directions of future work.

2. Equations, geometry, and basic solver

The basic equation set for the plasma is a two-dimensional (2D), toroidally axisymmetric fluid model that describes the evolution of the ion density, ion parallel momentum, electron temperature and ion temperature. The neutral species are described by two

possible fluid models: one in which the neutral parallel velocity is computed by including only charge-exchange coupling to ions and the neutral pressure gradient, and the second in which neutral parallel inertia and viscosity are included. For either neutral model, the ion and neutral temperatures are assumed strongly coupled through charge-exchange collisions and are thus described by a common "ion" temperature. Both plasma and neutral equations have strong nonlinearities representing convective/diffusive transport and coupled source/sink terms, as shown in Section 2.1.

Efforts are also increasing in kinetic edge plasma transport models (XGC, COGENT) [16, 17] that include two added velocity space dimensions to describe the plasma particle distribution functions. Also, kinetic Monte Carlo neutral codes (EIRENE, DEGAS2) [18, 19] are sometimes used, but these kinetic models are not considered here. One of the practical differences between kinetic and fluid transport simulations is the timescale: fluid transport simulations can more easily be evolved over longer time scales, making them more efficient for steady-state simulations, though not as rich in physics content.

An early example of implementation of a 2D fluid edge transport model is the B2 code [20, 21], which uses the semi-implicit SIMPLE algorithm [22] to solve the Navier-Stokes neutral fluid equations. More recently, B2 has evolved to SOLPS [23], now using a implicit iterative scheme in which subsets of equations are solved successively.

The original uniprocessor approach taken by UEDGE [24] was to solve the complete system of equations fully implicitly by using a preconditioned Newton-Krylov method. A numerical finite-difference preconditioning Jacobian was formed and approximately inverted by using ILUT [25]. Subsequently, work was performed on a domain-decomposed parallel algorithm for the plasma equations by using a full finite-difference Jacobian within each domain but with no preconditioner coupling between domains [8].

2.1. Physical model and geometry

Edge plasma transport can be characterized by fluid moments of the underlying collisional kinetic equation for a plasma in a strong magnetic field; a general description is found in [26], or more recently [27]. Here studies are conducted using equations for plasma particle continuity, parallel momentum density, separate ion and electron energy

densities, and neutral particle continuity as described in [7]:

$$\frac{\partial}{\partial t}(n_\mathrm{i}) + \nabla\cdot(n_\mathrm{i}\boldsymbol{v}_\mathrm{i}) = S_\mathrm{i}^P \tag{1a}$$

$$\frac{\partial}{\partial t}(n_\mathrm{i}m_\mathrm{i}v_{i\|}) + \nabla\cdot(n_\mathrm{i}m_\mathrm{i}v_{i\|}\boldsymbol{v}_\mathrm{i}) + \nabla_\|(n_\mathrm{i}T_\mathrm{i}) = en_\mathrm{i}E_\| - (\nabla\cdot\Pi_\mathrm{i})_\| - R_{\mathrm{i}\|} - m_in_n\nu_{cx}(v_{i||} - v_{n||}) + S_\mathrm{i}^m \tag{1b}$$

$$\frac{\partial}{\partial t}\left(\frac{3}{2}n_\mathrm{i,e}T_\mathrm{i,e}\right) + \nabla\cdot\left(\frac{5}{2}n_\mathrm{i,e}T_\mathrm{i,e}\boldsymbol{v}_\mathrm{i,e}\right) - \boldsymbol{v}_\mathrm{i,e}\cdot\nabla(n_\mathrm{i,e}T_\mathrm{i,e}) = -\nabla\cdot\boldsymbol{q}_\mathrm{i,e} - \Pi_\mathrm{i,e}\cdot\nabla\boldsymbol{v}_\mathrm{i,e} + Q_\mathrm{i,e} + S_\mathrm{i,e}^E \tag{1c}$$

$$\frac{\partial}{\partial t}(n_\mathrm{n}) + \nabla\cdot(n_\mathrm{n}\boldsymbol{v}_\mathrm{n}) = K_H^\mathrm{i}n_\mathrm{e}n_\mathrm{n} - K_H^\mathrm{r}n_\mathrm{i}n_\mathrm{e} \tag{1d}$$

$$\frac{\partial}{\partial t}(n_\mathrm{n}m_\mathrm{n}v_{n\|}) + \nabla\cdot(n_\mathrm{n}m_\mathrm{n}v_{n\|}\boldsymbol{v}_\mathrm{n}) + \nabla_\|(n_\mathrm{n}T_\mathrm{n}) = -(\nabla\cdot\Pi_\mathrm{n})_\| + m_in_n\nu_{cx}(v_{i||} - v_{n||}) + S_\mathrm{n}^m. \tag{1e}$$

We analyze cases using deuterium (denoted D, having a proton-neutron nucleus with a single electron), typical of most present-day fusion devices. The primary variables evolved are n_i and n_n for D^+ and D^0 densities, $v_{i\|}$ and $v_{n\|}$ for ion and neutral velocities along $\mathbf{B}$, $T_i = T_n$ for the common ion/neutral temperature, and T_e for the electron temperature. The parallel electric field, $E_\|$, can be replaced with plasma variables by using the electron parallel momentum equation, neglecting inertia and electrical current:

$$eE_\| = -T_e\nabla_\|(\ln n_e) - 1.71\nabla_\|(T_e), \tag{2}$$

where $n_e = n_i$ is the electron density in this quasi-neutral plasma. The simplest neutral model neglects inertial and viscous terms in Eq. (1e), in which case $v_{n\|}$ can be computed algebraically from other variables, yielding a five-variable equation set used for some of the calculations in later sections.

Physical terms and parameters in the equations just given include S_i^p - source for neutral ionization and recombination, e - elementary charge, $m_{i,n}$ for ion and neutral masses, $\Pi_\mathrm{i,e}$ - viscous force, $R_{\mathrm{i,e}\|}$ - friction force, $\boldsymbol{q}_\mathrm{i,e}$ - heat fluxes, $Q_\mathrm{i,e}$ - volume heating terms, $S_\mathrm{i,e}^E$ - external source terms, K_H^i - ionization rate coefficient, and K_J^r - recombination rate coefficient. The term ν_{cx} is the ion/neutral charge-exchange rate. The parallel transport coefficients come from collisional theory [26], whereas the perpendicular transport coefficients arise from plasma turbulence and are typically deduced from comparison of outer midplane profiles with experimental data.

The edge-region domain for a magnetically diverted tokamak has an annular shape as shown in Figure 1(a). A strong toroidal magnetic field, B_t, and a weaker poloidal magnetic field, B_p, are used to confine the hot core plasma. The resulting field lines form closed surfaces in the (red) inner edge region and the hot core, but are open outside the dotted separatrix where much of the plasma exhaust flux is directed to material divertor plates.

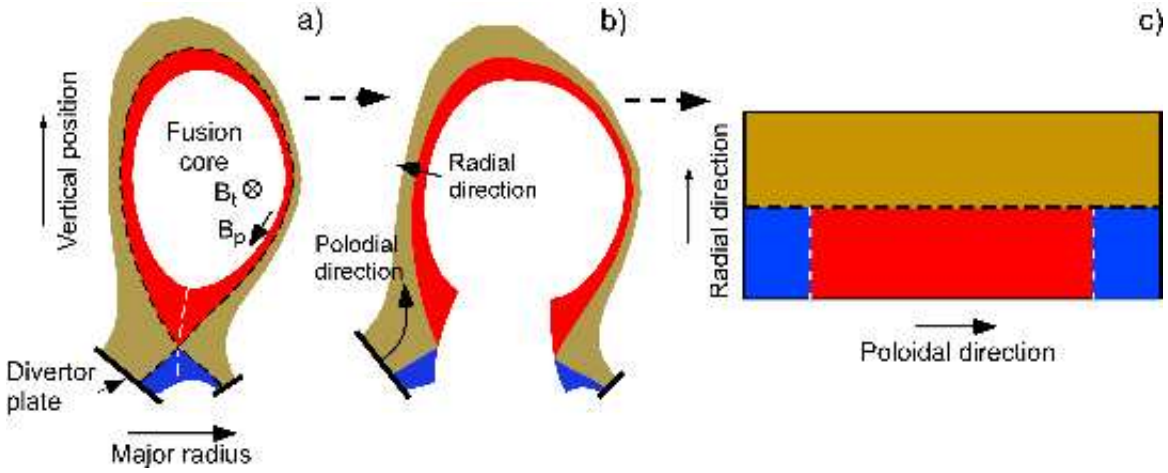


Figure 1: Conceptual steps are shown for viewing (a), the original edge tokamak geometry in poloidal cross-section as (c), a logically rectangular domain. The original domain is cut along the white dotted line in (a), opened as in (b), and reformed into the equivalent domain (c), where the white dotted line now appears as two internal boundaries. The dotted black line denotes the magnetic separatrix dividing regions of closed magnetic field lines in the core and open elsewhere.

Boundary conditions are applied to the second order differential set in Eqs. (1) based on physical considerations. A more detailed discussion is available [7]. In summary, on the boundaries touching the hot core plasma or the side walls, either Dirichlet or Neumann boundary conditions can be applied to the densities and temperatures, specifying the variable value or its flux into or out of the simulation volume. Here we use Dirichlet conditions, and the numerical behavior is not sensitive to this choice. For the parallel velocities, on the core or outer wall, a radial slip condition is used, *i.e.*, zero normal derivative of the velocities. For the inner/outer divertor plate boundaries, the magnetic field lines intersect the material surfaces with strong plasma particle and energy flows onto the plates. The ion and electron temperature equations use mixed Robin boundary conditions to describe the flow of heat. The ion parallel velocity is set to the ion acoustic speed with the neutral parallel velocity being a fraction of that of the ion. A Neumann condition is applied to the ion density. The neutral flux into the plasma is proportional to the D^+ incident flux; that is, $\Gamma_{D^0} = -R_p \Gamma_{D^{+1}}$. This particle recycling is a major process at the divertor plates (and walls) and is characterized by R_p near unity.

A quadrilateral-cell mesh is used for finite-volume discretization, with one coordinate being along magnetic flux surfaces and the other coordinate being orthogonal to the first or sometimes modified to fit along material surfaces; metric coefficients preserve the original geometrical features in the logically rectangular domain. The finite-volume spatial discretization stencil contains 9 points: the central cell of interest and the 8 surrounding cells adjacent to the center. The parallel domain partitioning can be either 1D, in the radial direction of Figure 1(c), or 2D, also including partitioning

in the poloidal direction. When considering the cost of parallel communication for evaluation of the nonlinear function, which requires data to be passed between adjacent domains, the 2D decomposition scales better than the 1D variant because the surface-area-to-volume ratio is lower, resulting in less communication.

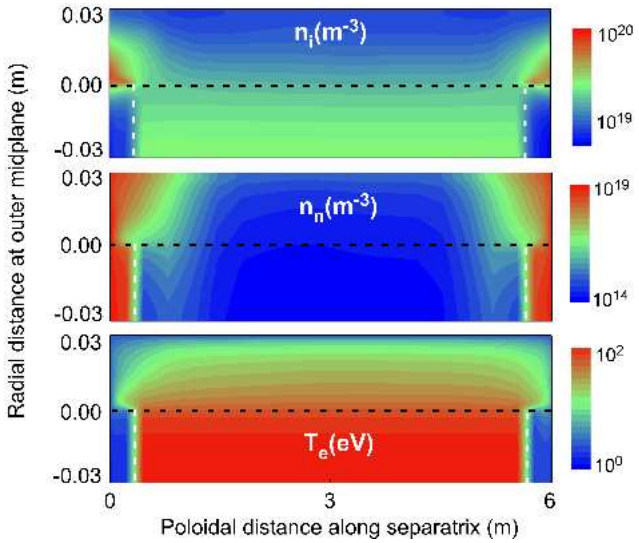


Figure 2: Typical steady-state profiles for ion density, neutral density, and electron temperature show very different structure. Figure 1 shows how this configuration relates to the original toroidal geometry.

To illustrate the range of variable amplitiudes, Figure 2 shows a typical steady-state solution for this edge plasma/neutral system for ion density, n_i, neutral density, n_n, and electron temperature, T_e. In comparison with Figure 1, one can see the large neutral density near the divertor plates [left and right boundaries in panel (c)] that is rapidly attenuated by ionization in the central edge-plasma region. The detailed input parameters for Figure 2 are given in Section 3.

In order to arrive at the steady state shown in Figure 2, many time scales need to be stepped over. Figure 3 shows some of the important time scales for tokamak edge plasma/neutral systems.

The plasma transport is that of an anisotropic fluid, where the plasma *fluid* is charged and thus strongly affected by the external magnetic field. The plasma equations for density (1a), momentum (1b), and energy (1c) all govern variables that can be characterized as anisotropic fluids. Ion flow in the poloidal direction is generally rapid compared with the radial direction owing to a component of the large parallel velocity mapping to the poloidal coordinate, which is the essence of the the magnetic confinement mechanism. In fact, there is a large anisotropy in all the plasma transport coefficients in these two directions that can be illustrated by considering classical thermal transport, $\chi_{\parallel}$ and $\chi_{\perp}$. Along the magnetic field, Coulomb collisional mean-free path, λ, defines the diffusion distance, whereas across the field, it is the gyroradius, ρ. Thus, the classical

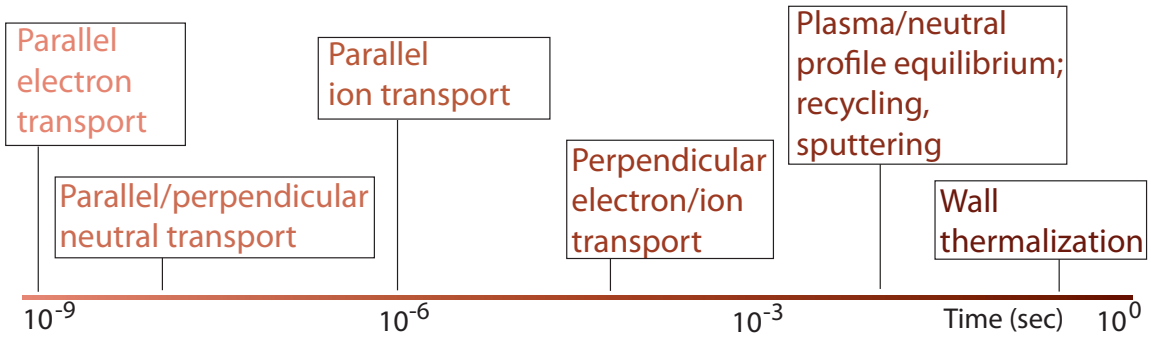


Figure 3: Significant phenomena in an edge simulation and their associated time scales. Of particular note is the large gap between perpendicular neutral diffusion and perpendicular ion diffusion.

parallel-to-perpendicular thermal diffusivity ratio scales as

$$\frac{\chi_\perp}{\chi_\parallel} \sim \left(\frac{\rho}{\lambda}\right)^2.$$

This ratio can be very small in the tokamak edge plasma region, especially for electrons where values of $\rho/\lambda \sim 10^{-4} - 10^{-5}$ can occur. However, radial transport is typically dominated by plasma turbulence rather than the classical collision process. Consequently, $\chi_{e\perp}$ can be enhanced by a few orders of magnitude, but the anisotropy remains large.

In contrast to the plasma, the neutral particles, governed by Eq. (1d), diffuse isotropically because they are unaffected by the external electric field. This neutral isotropy has a significant effect on the numerical conditioning of the system because the mesh is designed in deference to the plasma terms and is thus highly anisotropic. Section 4.1 analyzes this conditioning issue.

Extending beyond the basic deuterium plasma, an impurity species such as neon can be added to the simulation to model reduction of the peak heat flux to materials via radiation losses. Electron impact excitation and ionization on neon, as well as electron-ion recombination, result in a net energy loss to electrons via radiation loss (escaping photons). In the core region such radiation degrades fusion power gain. In the edge region, however, this radiative energy loss can be beneficial by distributing the plasma exhaust power over a wide area on the wall, thus reducing the focused peak heat load on divertors [28]. Neon increases the number of independent fluid variables by 11 per cell because of the additional 10 neon charge states (Ne^{+1} through Ne^{+10}) and the neon atomic neutral species.

2.2. Nonlinear solver

The UEDGE equations are spatially discretized with a finite volume approach [29], and the implicit Euler scheme is used to discretize time-dependent terms [30]. The coupled

system of equations (1a)-(1e) are of the form

$$\frac{d\boldsymbol{u}}{dt} = G(\boldsymbol{u})$$

for some complicated function G. Here $\boldsymbol{u}$ is a vector composed of the unknown variable values: at a minimum, plasma velocity, plasma density, ion temperature, electron temperature, and neutral density. Applying an implicit Euler discretization produces a discrete nonlinear equation

$$F(\boldsymbol{u}_k) = \frac{\boldsymbol{u}_k - \boldsymbol{u}_{k-1}}{\Delta t} - G(\boldsymbol{u}_k) = 0, \tag{3}$$

which must be solved at each time step Δt.

The choice of time step depends on several factors. If the primary interest is in the steady-state solution to the transport problem, as large a time step as possible is taken, where for some cases Δt can be arbitrarily large such that a nonlinear algebraic system of equations is solved iteratively. If such a direct steady-state solution is not possible, however, a time step can aid convergence by adding substantial terms on the diagonal of the preconditioning matrix. One can then progressively increase the time step to obtain a steady-state solution without resolving the full dynamics of the system. The final case is where the dynamics are of interest, and Δt should then be small enough to resolve the associated temporal evolution of the system. Here we are interested primarily in the steady-state solutions, so the time step is used as an aid to convergence of the system of equations.

We employ a Newton method (see, e.g., [31]) to solve Eq. (3), which is an iterative method consisting of two parts: a linear solve and a vector update. Labeling $\boldsymbol{u}_k$ the unknown at the k^{th} time step, these parts can be expressed as

$$\begin{aligned}
&\text{For } n = 0, 1, ..., N \\
&\quad \mathsf{J}(F)(\boldsymbol{u}_k^n)\Delta\boldsymbol{u}_k^{n+1} = -F(\boldsymbol{u}_k^n) && \text{(4a)} \\
&\quad\quad \boldsymbol{u}_k^{n+1} = \boldsymbol{u}_k^n + \Delta\boldsymbol{u}_k^{n+1}, && \text{(4b)}
\end{aligned}$$

where $\boldsymbol{u}_k^n$ is the result of the n^{th} iteration, and $\mathsf{J}(F)(\boldsymbol{u})$ is the Jacobian of F evaluated at $\boldsymbol{u}$. Once convergence is reached at step N, the next time step is set as $\boldsymbol{u}_k = \boldsymbol{u}_k^N$. The initial guess for a given time step is $\boldsymbol{u}_k^0 = \boldsymbol{u}_{k-1}$, where $\boldsymbol{u}_1^0$ is the initial solution.

For practical applications, (4b) is not the actual update used at each Newton step. Instead, a line search [31] is used to damp the magnitude of $\Delta\boldsymbol{u}_k^{n+1}$ to prevent oscillatory convergence behavior common during nonlinear solves. This changes the update portion of the nonlinear solve from (4b) to

$$\boldsymbol{u}_k^{n+1} = \boldsymbol{u}_k^n + \alpha^{n+1}\Delta\boldsymbol{u}_k^{n+1}, \tag{5}$$

where $0 < \alpha_{min} < \alpha^{n+1} \leq 1$ is chosen according to some scheme. Experiments here use a cubic interpolation scheme described in [32].

The final change for Newton-Krylov solvers in practical implementations comes in the linear solve. In [33] it was determined that the quadratic convergence associated with Newton methods can be preserved even when (4a) is solved only approximately. This revelation opens nonlinear solvers to the possibility of solving (4a) with an iterative method, thus introducing the *Krylov* half into the term Newton-Krylov method. The iterative method employed here is GMRES [34], which is appropriate in this context because $\mathsf{J}(F)$ is neither symmetric nor positive definite, and $\mathsf{J}(F)^T$ is not available.

When a Newton-Krylov method is performed in the absence of the true Jacobian, it is often called a matrix-free or Jacobian-free Newton-Krylov (JFNK) approach [35]. By using finite differences, the matrix-vector product $\mathsf{J}(F)(\boldsymbol{u})\boldsymbol{b}$ can be approximated:

$$\begin{aligned} F(\boldsymbol{u}+\delta\boldsymbol{b}) &= F(\boldsymbol{u}) + \delta\mathsf{J}(F)(\boldsymbol{u})\boldsymbol{b} + O(\delta^2)\frac{\boldsymbol{b}}{\|\boldsymbol{b}\|} \\ \iff \mathsf{J}(F)(\boldsymbol{u})\boldsymbol{b} &\approx \frac{F(\boldsymbol{u}+\delta\boldsymbol{b}) - F(\boldsymbol{u})}{\delta}. \end{aligned} \tag{6}$$

By computing finite difference Jacobian-vector products for the iterative solve, assembling the actual Jacobian is never required. Rather, some approximation to the inverse of the Jacobian can be used to precondition (4a). Choosing an effective approximation is one of the goals of this work.

Note that the older serial version of UEDGE has used a version of the Newton-Krylov solver NKSOL [36] with the first-order $1/\Delta t$ Euler term of Eq. (3) added. UEDGE has also previously used the PVODE [37] solver with a higher-order Δt algorithm that is now a component in the SUNDIALS [38] package. Some initial parallel results results for UEDGE in [8] utilized PVODE. However, all results in the present paper use the backward Euler discretization with nonlinear solvers from PETSc [39], which provides parallel sparse Jacobian computation via finite differences with coloring [40], matrix-free Jacobian-vector products, suites of Krylov methods and preconditioners, and consistent profiling [41].

Some preconditioning clarification is needed for this work. The term LU preconditioner means the full LU decomposition [34] applied to a Jacobian approximation. In contrast, the additive Schwarz method (ASM) preconditioner [42] is based on domain decomposition and conducts local solves (in this case, we use local LU factorization) on only the part of the domain owned by each processor. A tuning parameter called the overlap, when increased, allows for more communication between processor nodes at additional cost. The default of 1 cell overlap will be used in all experiments unless otherwise noted; for these UEDGE models, there was little or no improvement in the solve speed for larger overlap. Note that when using a single processor ($NP = 1$), these two preconditioners are actually the same. We also consider ILU(k) [25], symmetric successive over-relaxation (SSOR) [34], algebraic multigrid (AMG) [43], and ILUdt [44], an incomplete LU factorization utilizing a drop tolerance strategy, as discussed in the following sections.

3. Results using basic solver/preconditioner

Having described the plasma transport simulation and the structure of the associated solver, we now analyze its efficiency for typical edge plasma parameters. The geometry corresponds to the DIII-D tokamak as shown in Figure 1. The initial conditions for these simulations are taken from a steady-state solution obtained for core-boundary input parameters of fixed $T_e = T_i = 100$ eV and $n_i = 2.5 \times 10^{19}$ m^{-3}. The radial transport coefficients for all plasma variables are set to 1 m^2/s for simplicity, a value that is in the range typically deduced for present-day devices. The divertor plate particle recycling coefficient is set to $R_p = 0.9$. For the simulations presented in this section, the core-boundary values of T_e and T_i are then raised to 120 eV, and the new system is solved to a convergence level where the L2-norm of the residual is reduced by a factor of 10^8 from the initial residual for a given time step.

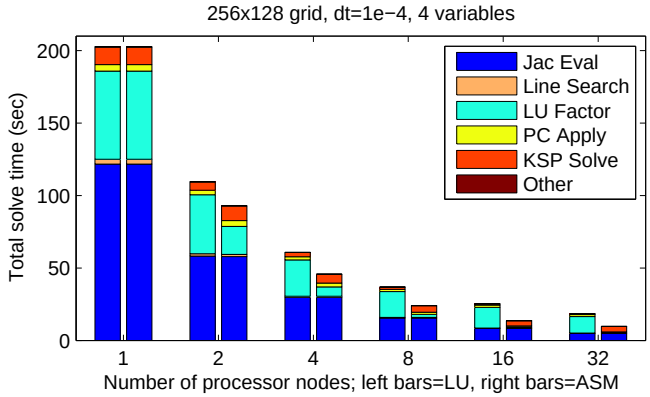
The first subsection shows the effectiveness of basic preconditioners for obtaining good multiprocessor performance when only the plasma variables are evolved. The next subsection illustrates the impact on parallel scaling when the neutral density evolution is added to that of the plasma.

3.1. Solver performance for fixed neutral gas

Because the computational grid is laid out anisotropically in deference to the dominant plasma terms, there should be a reasonably conditioned problem when the plasma terms are handled separately and the neutral terms fixed. The timing results for "plasma only" simulation on a 256x128 grid are displayed in Figure 4; the preconditioners being considered are the LU preconditioner and the ASM preconditioner with 1 block per processor, overlap of 1 cell, and LU as the subdomain solver for each block.

NP	Speedup	
	LU	ASM
2	1.85	2.18
4	3.33	4.43
8	5.50	8.44
16	8.01	14.83
32	11.03	20.94

(a) The ASM preconditioner scales significantly better than the LU preconditioner.



(b) The **LU Factor** bar for the LU preconditioner (the left bars) stagnates and dominates the computational cost for larger NP.

Figure 4: Results showing that plasma transport simulations (with fixed neutral terms) are well conditioned because they can be easily solved with the weaker and more scalable ASM preconditioner.

The bar graphs such as Figure 4b, show the proportion of time spent in various phases of the nonlinear solve; these are as listed below, along with an italicized comment explaining whether that cost depends on the number of linear iterations required. For instance, Jacobian evaluation occurs once at the beginning of each nonlinear iteration, so its cost is *linear iteration independent*.

- **Jac Eval** - *Linear iteration independent* - Evaluation of the approximate Jacobian used for preconditioning only.
- **Line Search** - *Linear iteration independent* - Improves most of the globalization of JFNK by reducing the size of the suggested Newton step (see Section 2.2).
- **LU Factor** - *Linear iteration independent* - Production of triangular factors. For the LU preconditioner, this refers to the global factorization; for the ASM preconditioner, this refers to local factorizations on each processor.
- **PC Apply** - *Linear iteration dependent* - Preconditioning linear solve using factors from **LU Factor**.
- **KSP Solve** - *Linear iteration dependent* - **K**rylov **S**ubs**P**ace (KSP) methods; in this problem GMRES is used; this phase includes Jacobian-vector products.
- **Other** - Convergence monitoring, incidental memory allocations, other trivial costs.

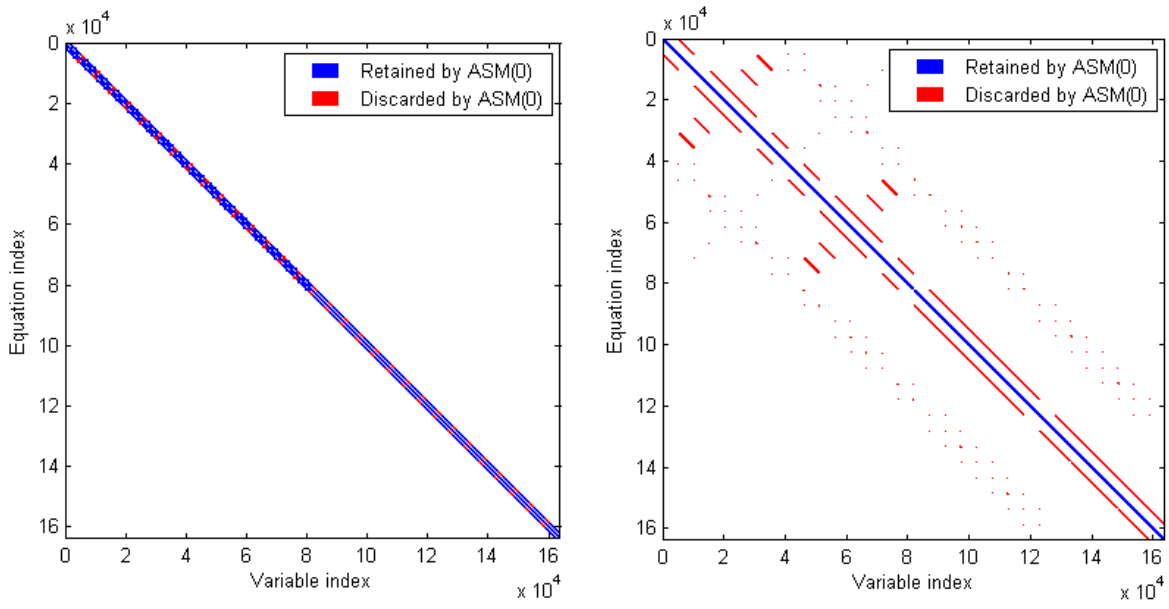
The ASM preconditioner is effective when the 1D domain decomposition is used because all the couplings in each radial strip are retained, while most couplings across radial strips are neglected. Less plasma transport information is lost when using the 1D decomposition (because fewer relevant terms cross the radial partition); therefore, the quality of the preconditioner remains high, and the solution is obtained more quickly. See Figure 5 for a comparison of the matrix nonzero patterns with 1D/2D domain partitions.

ASM is less effective than LU on ill-conditioned systems, so its success here shows that the "plasma terms only" problem is well conditioned with $\Delta t = 10^{-4}$. The choice of time step does affect the conditioning of the system. As discussed in [45], when problems of the form (3) are solved with a Newton iteration, the associated linear systems will have improved conditioning with a smaller time step Δt. This can be seen for a simple implicit time discretization scheme (like that of UEDGE), which produces a linearized system of the form (4a),

$$\underbrace{\left(\mathsf{I} - \Delta t\, \mathsf{J}(F)\left(\boldsymbol{u}_k^{(n)}\right)\right)}_{=\mathsf{A}} \Delta \boldsymbol{u}_k^{(n)} = \boldsymbol{u}_k^{(n)} - \boldsymbol{u}_{k-1} - \Delta t\, F\left(\boldsymbol{u}_k^{(n)}\right),$$

that needs to be solved iteratively. As is well known [46], a linear system is best solved when the spectrum of the system matrix, A, has eigenvalues clustered away from 0.

Assuming the Jacobian term $\mathsf{J}(F)\left(\boldsymbol{u}_k^{(n)}\right)$ is ill-conditioned (were it not, this problem would be easily solved), it has eigenvalues that are spread over a large region of the complex plane. By multiplying these eigenvalues by a small Δt term, the region of the complex plane covered by the eigenvalues is compressed toward the origin.



(a) 1D nonzero pattern

(b) 2D nonzero pattern

Figure 5: The 1D and 2D decompositions produce different global matrix orderings. These matrices were produced with $NP = 32$.

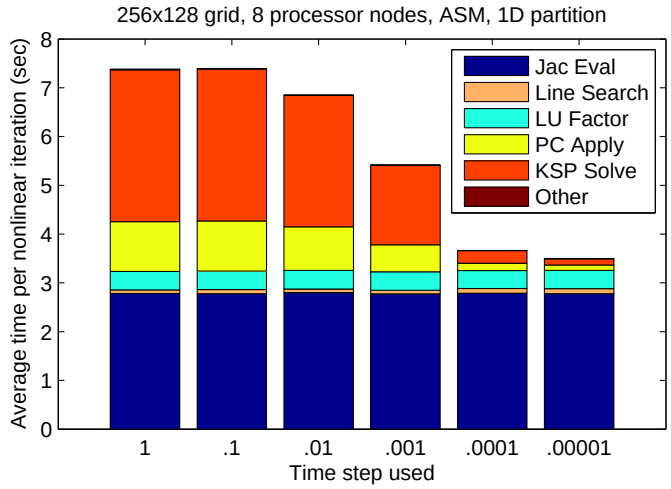
By itself, this would not cause any difference in the conditioning because both the largest and smallest eigenvalues would be scaled by the same factor and thus their ratio would remain unchanged. When A is actually formed, the identity matrix is added to the scaled Jacobian, pushing this cluster of scaled eigenvalues away from 0. As $\Delta t \to 0$ the eigenvalues should approach a cluster around 1 on the real axis. Such a matrix is very well conditioned, and iteratively solving such a system should require little effort.

Figure 6 shows this effect on the ASM preconditioner, which is not appropriate for an extremely ill-conditioned system. Its ineffectiveness is apparent in Figure 6b, where, for $\Delta t = 1$, 78 linear iterations are required on average for each nonlinear iteration. That number decreases to 4 linear iterations on average for $\Delta t = 10^{-5}$, which in turn cuts the average nonlinear iteration time in half.

As the quality of the preconditioner improves, the number of linear iterations required per nonlinear iteration should decrease. This situation is noticeable in Figure 6a, where the proportion of time spent on *linear iteration dependent* costs (**PC Apply** and **KSP Solve**) decreases significantly as Δt decreases.

When taking multiple time steps, the solve time implications may be more complicated than the situation described above. A decrease in Δt may change the number of nonlinear iterations required to converge because the initial guess at time t_n, $\boldsymbol{u}_n^{(0)}$, is chosen to be the solution from the previous time step $\boldsymbol{u}_{n-1}$. Basically, choosing a larger time step will likely yield a worse initial guess.

The cost of constructing the Jacobian (**Jac Eval**) is a substantial fraction of the average time per time step in Figure 6a, especially at small Δt. In order to reduce the



(a) Linear solve proportion decreases with time step.

Time step (sec)	Average linear iterations
1	78
.1	78
.01	67
.001	41
.0001	7
.00001	4

(b) Average linear iterations per nonlinear iteration decrease with time step.

Figure 6: As the time step decreases, the conditioning of the system improves, and the ASM preconditioner becomes more effective.

average cost for multiple time steps, it can be useful to lag Jacobian computations [35] and reuse Jacobians within and across time steps. In fact, this strategy has been used successfully for multiple time-stepping in serial UEDGE since its inception [24]. While a time step $\Delta t \in [10^{-4}, 10^{-3}]$ is appropriate for a coupled core-edge simulation or to observe fast time-scale dynamics within the edge, steady-state problems have a final time $T = O(1)$. The lagged-Jacobian approach can thus allow a steady state solution to be efficiently obtained by a succession of variable time steps, although this topic is not discussed further in this paper.

3.2. Impact of dynamic neutrals

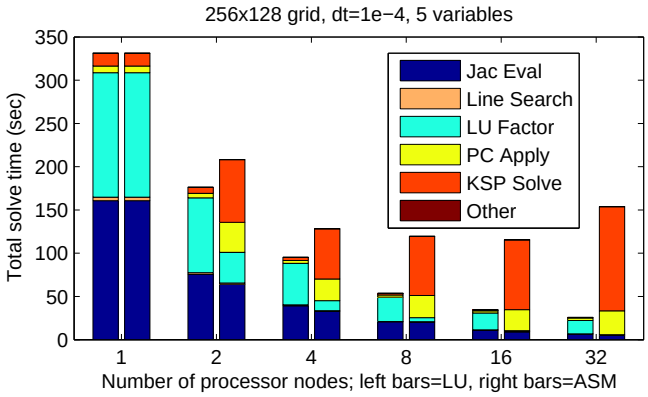
As has been documented [34], the full LU decomposition loses scalability because it is inherently global, and the additional nonzeros generated during the factorization require substantial interprocessor communication. This lack of strong scalability is displayed in Figure 7a, where the speedup profile looks unchanged compared with that in Figure 4a. For the original plasma-only simulations, this was not a problem, because we saw in Figure 4 that the ASM preconditioner scaled much better. Unfortunately, Figure 7 shows that is not the case for the coupled plasma/neutral simulation.

Figure 7b shows that the time required for performing the LU factorization fails to scale as the number of processors increases. For the LU preconditioner, each of the components of the nonlinear solve either scales well (**Jac Eval**) or becomes inconsequential compared with the cost of the solve *except* the **LU Factor** bar. The ASM preconditioner is not scaling well in this case, in direct contrast to the plasma-only simulation. Here the quality of the preconditioner degrades too significantly, and consequently too many GMRES iterations are required to reach convergence.

The choice of preconditioner is clearly a primary factor for scalability; neither the

NP	Speedup	
	LU	ASM
2	1.88	1.59
4	3.48	2.58
8	6.17	2.77
16	9.61	2.87
32	12.82	2.15

(a) The LU preconditioner maintains a similar speedup trajectory as in Figure 4a.



(b) The ASM preconditioner fails to scale almost immediately because the time spent in GMRES grows; this is caused by an increase in linear solver iterations.

Figure 7: The ASM failure shows that the coupled plasma/neutral transport simulation is very ill-conditioned. As expected, the LU preconditioner is indifferent to the change in conditioning caused by adding the neutral terms; hence, these results mirror those of the plasma transport simulation shown in Figure 4.

LU nor ASM preconditioners may be ideal for larger problems. As was determined in Section 3.1, the plasma terms alone are well conditioned, but introducing the neutral terms hinders the conditioning. Further analysis is required to understand these issues in preconditioning and to potentially improve scalability.

4. Improved decomposition algorithm for the plasma/neutral system

This section shows that isotropic diffusion on an anisotropic grid is intrinsically ill-conditioned. We develop a new preconditioner to isolate the troublesome neutrals from the plasma terms, and we demonstrate scalability improvements for several cases.

4.1. Analysis of neutrals on an anisotropic mesh

To understand the impact of adding the neutral component on the parallel scaling, we first consider the properties of the neutral continuity equation. For a high-density plasma typical of the edge plasma region, the neutral velocity is determined largely by charge-exchange collisions, namely, $\mathbf{v}_n \sim -\nabla(T_n/m_n)n_n\tau_{cx}$, where $T_n = T_i$ is the neutral temperature, and $\tau_{cx} = 1/\nu_{cx}$ is the neutral/ion charge-exchange time. Consequently, the neutral particle continuity equation (1d) has the form of a simple isotropic diffusion equation.

We consider a simple rectangular grid shown in Figure 8 with $\Delta y = \alpha\Delta x$ for $0 < \alpha \ll 1$. The governing PDE is the following Poisson equation with homogeneous

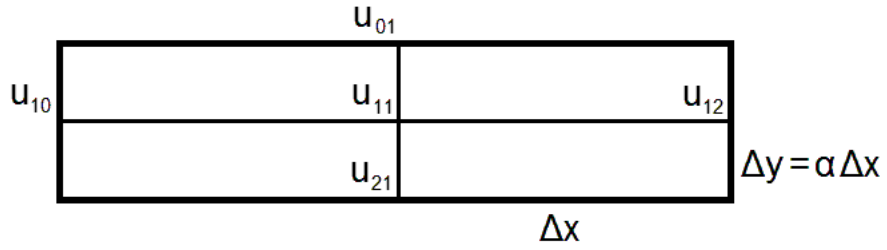


Figure 8: Grid spacing for Δx is much larger than Δy.

Dirichlet boundary conditions

$$\begin{aligned} \left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right) u(x,y) &= f(x,y) && \text{interior,} \\ u(x,y) &= 0. && \text{boundary} \end{aligned}$$

Using finite differences (for simplicity; UEDGE uses finite volumes) to discretize the PDE on the anisotropic grid from Figure 8 produces the system of equations

$$\begin{aligned} \frac{-2u_{ij}+u_{i+1\,j}+u_{i-1\,j}}{(\Delta x)^2}+\frac{-2u_{ij}+u_{ij+1}+u_{ij-1}}{(\Delta y)^2} &= f_{ij} && i=j=1 \\ u_{ij} &= 0 && \text{otherwise,} \end{aligned}$$

where f_{11} is f evaluated at the same location as u_{11}. Because of the structure of this particular stencil, the corner points do not affect the interior point, so their value need not be included in the solution of the system. Their omission leaves a system of 5 unknowns, which after substituting $\Delta y = \alpha \Delta x$ takes the $\mathsf{A}\boldsymbol{x} = \boldsymbol{b}$ form

$$\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ \alpha^2 & 1 & 1 & \alpha^2 & -(2+2\alpha^2) \end{pmatrix} \begin{pmatrix} u_{01} \\ u_{10} \\ u_{12} \\ u_{21} \\ u_{11} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \alpha^2(\Delta x)^2 f_{11} \end{pmatrix}.$$

This particular system can be easily inverted to $\boldsymbol{x} = \mathsf{A}^{-1}\boldsymbol{b}$

$$\begin{pmatrix} u_{01} \\ u_{10} \\ u_{12} \\ u_{21} \\ u_{11} \end{pmatrix} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ \frac{\alpha^2}{2+2\alpha^2} & \frac{1}{2+2\alpha^2} & \frac{1}{2+2\alpha^2} & \frac{\alpha^2}{2+2\alpha^2} & \frac{-1}{2+2\alpha^2} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \alpha^2(\Delta x)^2 f_{11} \end{pmatrix},$$

giving the 1-norm condition number of the system as

$$\begin{aligned} \kappa_1(\mathsf{A}) &= \|\mathsf{A}\|_1 \|\mathsf{A}^{-1}\|_1 \\ &= (2+2\alpha^2)\left(1+\frac{1}{2+2\alpha^2}\right) \\ &= 3+2\alpha^2. \end{aligned}$$

Since the restriction was made that $0 < \alpha \ll 1$, we have $\kappa_1(\mathsf{A}) \in (3, 5]$, which is a well-conditioned system. Initially that may seem like a positive outcome, but one must analyze the implications of that α restriction on the right hand. As $\alpha \to 0$, $\alpha^2(\Delta x)^2 f_{11} \to 0$ as well, which leaves a right-hand side that begins to look homogeneous as the grid anisotropy becomes more extreme. To compensate for this shrinking nonzero term, the matrix condition number should increase; instead it is bounded from above by 5, so that the system becomes insensitive to very small terms.

This issue is not caused simply by the choice of $\Delta y = \alpha \Delta x$ for $0 < \alpha \ll 1$ rather than $\Delta x = \gamma \Delta y$ for $\gamma \gg 1$. Reformulating the system with this discretization as $\hat{A}$ produces a similar matrix and a similarly structured condition number

$$\begin{aligned}\kappa_1(\hat{\mathsf{A}}) &= (2 + 2\gamma^2)\left(1 + \frac{\gamma^2}{2 + 2\gamma^2}\right) \\ &= 2 + 3\gamma^2.\end{aligned}$$

Unfortunately, an anisotropic grid is characterized in this formulation as $\gamma \to \infty$, which shows that the condition number is not bounded for this system but, rather, grows quadratically with γ. Thus, the system can be ill-conditioned as a direct result of discretizing an isotropic diffusion operator on an anisotropic grid.

4.2. The FieldSplit algorithm

Section 3.2 observed that domain decomposition based preconditioners such as ASM are not effective for solving the challenging coupled neutral and plasma species simultaneously on an anisotropic grid. However, ASM was successful when preconditioning the system with only the plasma terms in Section 3.1. This could be interpreted as part of the system being amenable to a cheaper preconditioner that proves ineffective when applied to the coupled system. Thus, we consider the coupled system as organized in the traditional block structure of one block of unknowns per grid point:

$$\underbrace{\begin{pmatrix} \begin{bmatrix} n_i \\ m_i v_\parallel \\ T_i \\ T_e \\ n_n \end{bmatrix}_1 \Bigg\} \boldsymbol{u}_1 \\ \vdots \\ \begin{bmatrix} n_i \\ m_i v_\parallel \\ T_i \\ T_e \\ n_n \end{bmatrix}_M \Bigg\} \boldsymbol{u}_M \end{pmatrix}}_{\text{unknowns}} \qquad \underbrace{\begin{pmatrix} \begin{bmatrix} \\ \mathsf{J}_1(F)(\boldsymbol{u}_1) \\ \\ \end{bmatrix} & \cdots & \begin{bmatrix} \\ \mathsf{J}_1(F)(\boldsymbol{u}_M) \\ \\ \end{bmatrix} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix} \\ \mathsf{J}_M(F)(\boldsymbol{u}_1) \\ \\ \end{bmatrix} & \cdots & \begin{bmatrix} \\ \mathsf{J}_M(F)(\boldsymbol{u}_M) \\ \\ \end{bmatrix} \end{pmatrix}}_{\text{Jacobian}}$$

where $\mathsf{J}_k(F)(\boldsymbol{u}_\ell)$ is the Jacobian of F with respect to the set of unknowns $\boldsymbol{u}_k$ evaluated at the set of unknowns $\boldsymbol{u}_\ell$, assuming M total grid points. Here the plasma terms are

colored red and the neutral terms blue. The vector of values $\boldsymbol{u}_\ell$ is colored green to signify that neither a preconditioner for the plasma nor neutral terms is fully effective.

If the vector of unknowns were reordered so that the neutral terms were all segregated from the plasma terms, the resulting ordering would produce the Jacobian

$$\underbrace{\begin{pmatrix} \left.\begin{bmatrix} (n_i)_1 \\ (m_i v_\parallel)_1 \\ (T_i)_1 \\ (T_e)_1 \\ \vdots \\ (n_i)_M \\ (m_i v_\parallel)_M \\ (T_i)_M \\ (T_e)_M \end{bmatrix}\right\} \boldsymbol{u}_P \\ \left.\begin{bmatrix} (n_n)_1 \\ \vdots \\ (n_n)_N \end{bmatrix}\right\} \boldsymbol{u}_N \end{pmatrix}}_{\text{unknowns}} \qquad \underbrace{\begin{pmatrix} \begin{bmatrix} & \\ & \\ & \\ & \\ & \mathsf{J}_P(F)(\boldsymbol{u}_P) \\ & \\ & \\ & \\ & \end{bmatrix} & \begin{bmatrix} & \\ & \\ & \\ & \\ & \mathsf{J}_P(F)(\boldsymbol{u}_N) \\ & \\ & \\ & \\ & \end{bmatrix} \\ \begin{bmatrix} & \\ & \mathsf{J}_N(F)(\boldsymbol{u}_P) \\ & \end{bmatrix} & \begin{bmatrix} & \\ & \mathsf{J}_N(F)(\boldsymbol{u}_N) \\ & \end{bmatrix} \end{pmatrix}}_{\text{Jacobian}}$$

where now $\mathsf{J}_P(F)(\boldsymbol{u}_N)$ is the Jacobian of F with respect to the plasma terms evaluated at the neutral variables $\boldsymbol{u}_N$, assuming again M total grid points. Notice that after this permutation all plasma variables (red) are segregated from all neutral variables (blue), and no mixed terms (green) need to be evaluated. The only interaction between plasma and neutral variables occurs in the off-diagonal blocks.

Reordering changes the sparsity structure of the matrix, which was already well banded in the 1D partition case as shown in Figure 5a. This change in sparsity structure can increase the time required to compute and apply a preconditioner based on the full LU factorization. For other preconditioners, however, if the nonzeros moved away from the diagonal were already largely neglected or unimportant, then terms retained would be sufficient for an efficient linear solve.

Applying this logic to the reordered Jacobian from this problem produces

$$\mathsf{J}(F)(\boldsymbol{u}) = \begin{pmatrix} \mathsf{J}_P(F)(\boldsymbol{u}_P) & \mathsf{J}_P(F)(\boldsymbol{u}_N) \\ \mathsf{J}_N(F)(\boldsymbol{u}_P) & \mathsf{J}_N(F)(\boldsymbol{u}_N) \end{pmatrix} \longrightarrow \mathsf{P}_{BJ} = \begin{pmatrix} \mathsf{J}_P(F)(\boldsymbol{u}_P) & \\ & \mathsf{J}_N(F)(\boldsymbol{u}_N) \end{pmatrix} \tag{7}$$

where P_{BJ} is a block Jacobi preconditioner void of the off-diagonal blocks present in the true Jacobian. This preconditioner offers two key benefits over a strictly algebraic preconditioner such as ILU(k) or the domain decomposition preconditioner ASM.

(i) By neglecting the off-diagonal coupling terms $\mathsf{J}_P(F)(\boldsymbol{u}_N)$ and $\mathsf{J}_N(F)(\boldsymbol{u}_P)$, the system is broken into smaller subproblems for preconditioning, each of which can potentially be handled by a different solver.

(ii) The plasma and neutral terms have now been segregated for more efficient preconditioning, though these remain fully coupled for the global problem via the Jacobian-free Newton-Krylov approach.

Combining these two ideas with the preconditioning results from Sections 3.1 and 3.2 provides a logical method of inverting the two diagonal blocks from P_{BJ}: $\mathsf{J}_P(F)(\boldsymbol{u}_P)^{-1}$ will be produced with additive Schwarz, and $\mathsf{J}_N(F)(\boldsymbol{u}_N)^{-1}$ will be computed with algebraic multigrid [43], which is optimal for isotropic diffusion.

As motivated by these challenges in robust and scalable linear solvers for coupled edge plasma/neutral transport simulations and other multiphysics applications [1], a relatively recent addition to PETSc is the `FieldSplit` class of preconditioners [47], designed for the flexible hierarchical construction of block solvers. The general `FieldSplit` infrastructure solves linear block systems using either block relaxation or block factorization (Schur complement approaches). Arbitrary square block systems are supported, with the user specifying only index sets to identify each block (which may overlap). Any algebraic solver can be used in each block, and algorithmic choices can be specified as runtime options. Additional applications using `FieldSplit` infrastructure include geodynamics [48], ice sheet dynamics [49], power networks [50], and mesoscale materials modeling [51].

4.3. Results for the FieldSplit algorithm

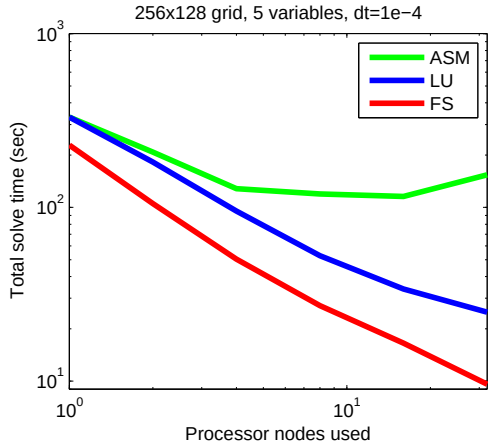
We implemented the preconditioning approach introduced in Section 4.2 using PETSc's `FieldSplit` (FS) infrastructure. This new approach to preconditioning — handling the plasma and neutral terms separately — improves both the solver speed and scalability, as shown in Figure 9. Note that we use the additive variant of FieldSplit, where the individual blocks are inverted separately and then the results are added together. For the remainder of this paper we reference only the additive approach when using the term FieldSplit or the abbreviation FS.

Gains made by the FieldSplit preconditioner over the global LU come from the reduced cost of the LU factorization within FieldSplit. Recall that the two disjoint blocks present in the preconditioner are inverted with two different methods:

- $\mathsf{J}_P(F)(\boldsymbol{u}_P)^{-1}$ is approximated by ASM with n domain partitions, where n is the number of processors. Each domain, along with a small overlap from neighboring domains, is then solved with LU factorization. Each LU factorization, computed on a block roughly $1/n$ times the dimension of $\mathsf{J}_P(F)$, requires no communication.
- $\mathsf{J}_N(f)(\boldsymbol{u}_N)^{-1}$ is computed by algebraic multigrid [52] (using hypre [53]) over the entire domain. Multigrid preconditioners are particularly effective for diffusion systems because of the orthogonal nature of the near and far eigenfunctions of the solution. Solving for the far range diffusion effects (the so-called coarse grid solve), as well as moving between grid levels, requires communication between processes and becomes less efficient as the number of processors increases.

Figure 9a shows the FieldSplit preconditioner consistently outperforming the LU preconditioner to the point where for $NP = 32$, FieldSplit is roughly 2.5 times faster.

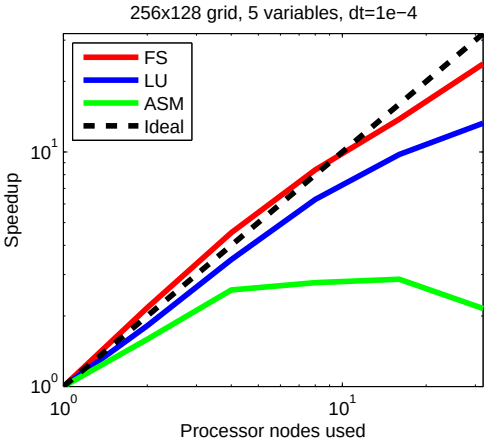
For the LU preconditioner, the cost of computing and applying the preconditioner scales poorly, as expected because of the level of fill-in, which requires additional memory



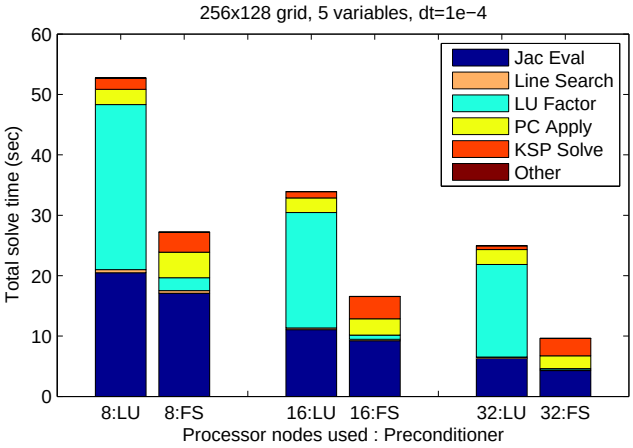
(a) FS solves are much faster than LU.

NP	Average linear iterations			Speedup		
	LU	ASM	FS	LU	ASM	FS
1	7	7	13	1.0	1.0	1.0
2	7	87	17	1.9	1.6	2.2
4	7	136	17	3.5	2.6	4.6
8	7	255	22	6.1	2.8	8.4
16	7	650	32	9.5	2.9	13.4
32	7	1585	44	12.7	2.1	22.8

(b) FS iterations are fewer and less costly.



(c) FS scales better than LU and ASM.



(d) We incur little increase in LU factorization cost.

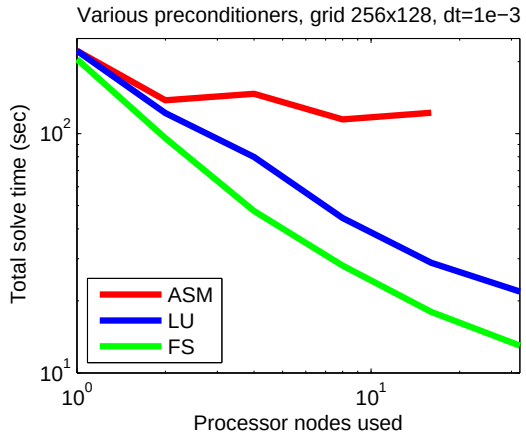
Figure 9: For the coupled plasma/neutral system, FieldSplit preconditioning outperforms LU at $\Delta t = 10^{-4}$.

allocation and interprocessor communication. Within FS, the cost of **LU Factor** drops dramatically, mostly because the sizes of the subdomains conducting those factorizations shrink. The increase in linear iterations per nonlinear iteration as seen in Figure 9b prevents the **PC Apply** and **KSP Solve** bars from scaling as well as they could. This corresponds then to a small loss in scalability for $NP = 32$ (as seen in Figure 9c) because the **LU Factor** bar is already as small as possible and the other linear solve terms are bounded by the number of linear iterations.

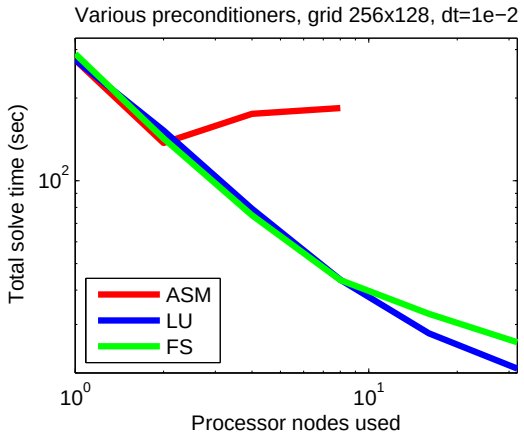
More qualitatively, FS can be thought of as a targeted preconditioner that uses more robust yet more costly techniques where necessary. The performance of this particular FS algorithm is predicated on being able to solve the plasma terms effectively with ASM even though Figure 9a shows that ASM scales poorly on the plasma/neutral coupled system. FieldSplit takes advantage of the results from Figure 6a, which show that ASM works well on the plasma terms, and uses a more costly yet more powerful solver (algebraic multigrid) on the tougher terms that prevent ASM from working well.

In conclusion, the significant improvement from LU preconditioning to FieldSplit can be attributed to selective use of a domain decomposition preconditioner on variables that do not demand global coupling during the solve.

4.3.1. Larger time steps Many significant tokamak dynamics can be observed with small time steps (refer to Figure 3), but in order to conduct a steady-state simulation, larger time steps may be appropriate. Unfortunately, as discussed in Section 3.1, some preconditioners fail to perform when applied to larger time steps. Figure 10 shows results for time steps 10 and 100 times larger than previous experiments.



(a) FieldSplit outperforms LU for $\Delta t = 10^{-3}$.



(b) FieldSplit falters near $\Delta t = 10^{-2}$.

NP	Preconditioner		
	LU	FS	ASM
1	7	25	7
2	7	26	52
4	8	22	103
8	8	33	240
16	8	43	621
32	8	53	**

(c) Average linear iterations per nonlinear iteration for $\Delta t = 10^{-3}$

NP	Preconditioner		
	LU	FS	ASM
1	7	39	7
2	7	46	51
4	6	44	180
8	6	56	427
16	6	89	**
32	6	113	**

(d) Average linear iterations per nonlinear iteration for $\Delta t = 10^{-2}$

Figure 10: FieldSplit performs adequately for $\Delta t < 10^{-2}$.
*Note: ** denotes divergence*

The LU preconditioner performs independently of Δt. The same cannot be said for the FieldSplit preconditioner, whose quality decreases with an increase in time step. This performance is in agreement with Section 3.1 and is likely brought on by the increasingly poor performance of ASM for larger timesteps (see Figures 10a and 10b).

These results bring to the forefront the significance of the ASM component of this particular FieldSplit preconditioner (as distinct from the more general PETSc `FieldSplit` infrastructure described in [47]). FieldSplit outperforms the LU

preconditioner when large portions of the system do not require the full LU factorization and thus are well conditioned. As the time step increases, the plasma terms become more difficult to solve, and the ASM preconditioner fails to provide an adequate speedup over the LU preconditioner. This situation may motivate future research on alternative physics-based solvers for portions of the variables, as well as a more general process for identifying appropriate FieldSplit components.

We also note that the quality of the initial guess will generally degrade given a larger time step. For this example there is no significant effect, but it is conceivable that the choice of initial guess could affect the convergence of FieldSplit given the longer path that each linear solve takes to convergence.

4.3.2. Including inertia and viscosity for the neutral parallel velocity A primary goal of studying different preconditioning techniques is to allow for flexibility in the solver for different physical simulations. In addition to the 5 degrees of freedom (5 variable types) that exist in numerical experiments thus far (D^+ temperature, D^+ velocity, D^+ density, electron temperature, D^0 density), the inclusion of neutral inertia and viscosity requires that the neutral parallel velocity equation be added to the system of differential equations; until now it has been computed algebraically by using the solution to the 5-variable differential system as described in Section 2. Doing so increases the share of equations describing the neutral terms from 20% to 33% (from 1 of 5 to 2 of 6).

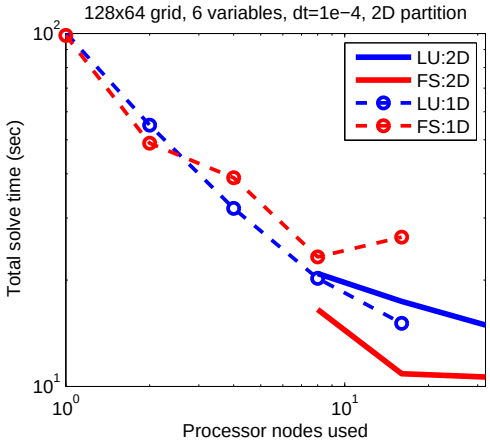
It is reasonable to assume that this shift will have an effect on the solve strategy. For 5 variables the 1D domain decomposition was preferred by the neutrals and thus was preferred by FieldSplit because only a single neutral term dissented. Doubling the number of neutral terms has shifted the balance of power, as can be seen in Figure 11, where the 2D partitioning is preferred to the 1D partitioning.

We note that 2D partitioning is not supported for the NP=4 case. To prevent a cell from having neighboring values both adjacent and across the branch cut, a 4-processor case is not viable. Also of note is that 3 fields are chosen in this FieldSplit preconditioner instead of 2 previously: plasma terms, neutral density, neutral momentum. This means that in the preconditioner, the terms coupling the neutral density with other variables are ignored, as well as the terms coupling the neutral momentum to other variables. In matrix form this looks like

$$\begin{pmatrix} \mathsf{J}_P(F)(\boldsymbol{u}_P) & \mathsf{J}_P(F)(\boldsymbol{u}_N) & \mathsf{J}_P(F)(\boldsymbol{u}_M) \\ \mathsf{J}_N(F)(\boldsymbol{u}_P) & \mathsf{J}_N(F)(\boldsymbol{u}_N) & \mathsf{J}_N(F)(\boldsymbol{u}_M) \\ \mathsf{J}_M(F)(\boldsymbol{u}_P) & \mathsf{J}_M(F)(\boldsymbol{u}_N) & \mathsf{J}_M(F)(\boldsymbol{u}_M) \end{pmatrix} \rightarrow \begin{pmatrix} \mathsf{J}_P(F)(\boldsymbol{u}_P) & & \\ & \mathsf{J}_N(F)(\boldsymbol{u}_N) & \\ & & \mathsf{J}_M(F)(\boldsymbol{u}_M) \end{pmatrix},$$

where $\boldsymbol{u}_M$ are momentum variables and $\mathsf{J}_M(F)$ is the Jacobian of F with respect to $\boldsymbol{u}_M$.

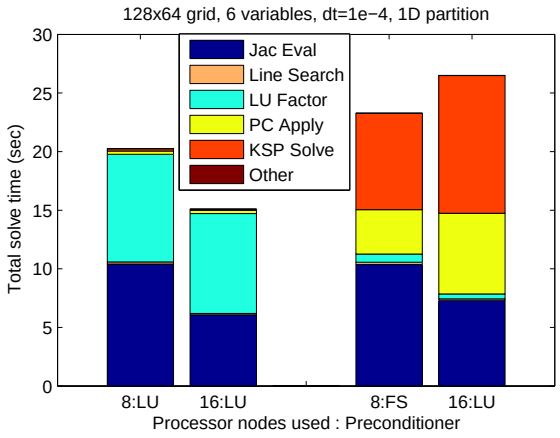
The main reason for the improved behavior with the 2D partition is the fewer linear iterations required per nonlinear iteration, as seen in Figure 11b. Because the $\mathsf{J}_N(F)(\boldsymbol{u}_N)$ and $\mathsf{J}_M(F)(\boldsymbol{u}_M)$ blocks are solved directly, the only difference between the 1D and 2D cases is the $\mathsf{J}_P(F)(\boldsymbol{u}_P)$ solve, which is conducted with ASM by using sequential LU on each subset of plasma terms. By ordering the plasma variables into 2D blocks rather than 1D strips, more significant terms are retained, making the preconditioner more effective.



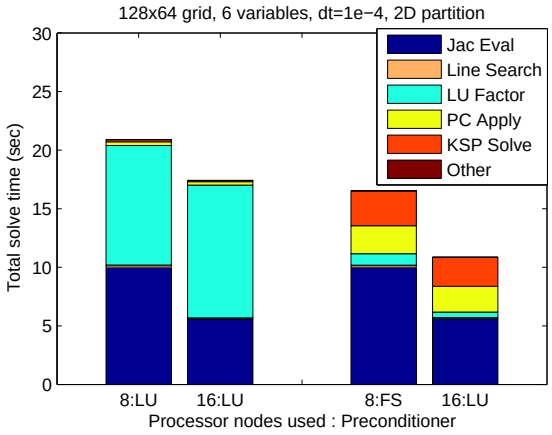
(a) 1D and 2D results differ significantly.

NP	Preconditioner		
	LU	FS	
		1D	2D
8	2	89	41
16	2	175	53

(b) Average linear iterations per nonlinear iteration



(c) In 1D, FieldSplit fails to perform.



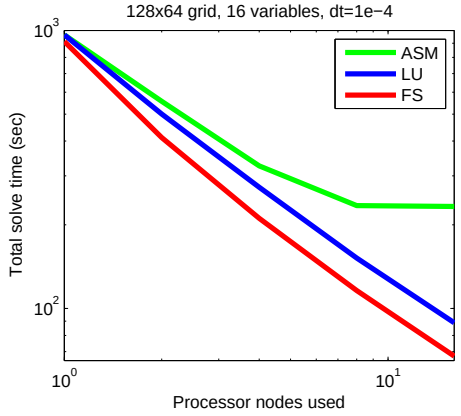
(d) In 2D, FieldSplit outperforms the LU preconditioner.

Figure 11: The 6-variable case sees better performance with FieldSplit using a 2D partitioning.

4.3.3. Adding a neon impurity Impurity seeding of a plasma discharge can be used to radiate a portion of the plasma energy and thus reduce the peak heat-flux to the divertor plates [28]. Introducing a neon impurity to the UEDGE simulation increases the complexity of the simulation because 10 new ion densities must be computed (one for each possible charge that a neon could take) as well as a neutral neon species. For a UEDGE simulation, this increases the total degrees of freedom at each grid point in the simulation to 16, with 5 original hydrogen variables and 11 neon variables; the neutral hydrogen and neon momentums with be solved for algebraically and not included in the system of differential equations. Also, for the neon cases we increase the initial core-boundary temperatures to $T_e = T_i = 400$ eV before perturbing them to 440 eV and increase the recycling coefficient to $R_p = 0.95$.

Using a 128x64 grid, we conducted experiments using LU, ASM, and FieldSplit preconditioners; the outcomes are compared in Figure 12. The FieldSplit performed

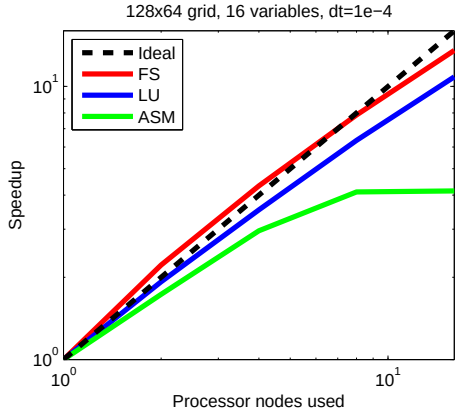
here resembles the structure from Section 4.2, where all the plasma terms (now including the neon ions) are segregated from the neutral terms (now including neutral neon). Figure 12b shows that both FieldSplit and LU incur almost no change in average linear iterations; this result, coupled with the diminishing cost of FieldSplit for more processors, explains the better scalability in Figure 12c. We note that on this smaller grid it is only possible to run up to $NP = 16$ for the 1D partition.



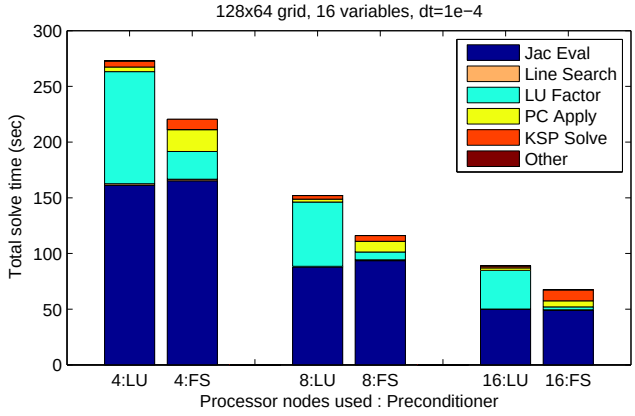
(a) FS solves are faster than LU.

NP	Average linear iterations			Speedup		
	LU	ASM	FS	LU	ASM	FS
1	8	8	21	1.0	1.0	1.0
2	8	79	19	1.9	1.7	2.2
4	8	128	21	3.5	3.0	4.3
8	8	281	29	6.3	4.1	7.9
16	8	622	51	10.8	4.1	13.6

(b) FS iterations are fewer and less costly, which allows for more speedup.



(c) FS scales better than LU and ASM.



(d) We observe different performance for FS and LU.

Figure 12: FieldSplit preconditioning outperforms LU when simulating a neon impurity.

We note that although this simulation involves more unknowns than the 5-variable (Section 4.3) and 6 variable (Section 4.3.2) cases, it is being run on the same size mesh. The greater degrees of freedom per node create a larger system with similar block structure, but larger blocks that are more dense. This is in contrast to a system with 5 active variables being run on a 256x128 mesh, which would have roughly the same size but a very different and sparser, nonzero pattern. Understanding the structure of the system is crucial to choosing an appropriate preconditioner, especially for FieldSplit, given the free parameters needed during its construction (selecting fields, choosing coupling, subsolvers/parameters).

5. Conclusions and future work

Coupled plasma/neutral edge transport simulations are vital to accurately predicting key design and operational issues for magnetic fusion energy devices (as well as other fusion concepts not treated here). However, models of edge systems are intrinsically ill-conditioned because of nonlinearities and disparate time scales. We have demonstrated that a physics-based preconditioner named FieldSplit, used with a Jacobian-free Newton-Krylov method, can provide an efficient method for solving coupled plasma/neutral simulations in parallel. Preserving strong coupling between the plasma and neutral terms is necessary for simulation fidelity, but traditional algebraic preconditioners are not generally scalable in this context because of the overhead of parallel communication. To improve efficiency and scalability of the solver, one can separate the FieldSplit preconditioner into neutral and plasma components, each consisting of its own physics and solved with an appropriate method. Specifically, good parallel scaling is obtained by preconditioning the troublesome neutral equations with algebraic multigrid and the plasma equations with an additive Schwarz preconditioner.

Preconditioning the individual components of a coupled problem while solving the fully coupled model with a Newton-Krylov method can be applied to more complicated problems in magnetic confinement fusion, including the addition of equations for the neutral deuterium momentum and/or an impurity species. Future work includes applying the FieldSplit approach to coupled core/edge simulations and investigating more advanced FieldSplit functionality (e.g., new component solves, Schur complements).

Acknowledgments

We thank Satish Balay, Charles Van Loan, and Barry Smith for insightful discussions. We gratefully acknowledge use of the Fusion cluster in the Laboratory Computing Resource Center at Argonne National Laboratory. T. D. Rognlien was supported by DOE contract number DE-AC52-07NA27344. L. McInnes and H. Zhang were supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. M. McCourt was partially supported by a National Science Foundation Graduate Research Fellowship and by DOE contract number DE-AC02-06CH11357. This work was largely carried out as part of the FACETS SciDAC project supported by DOE.

References

[1] D. E. Keyes, L. C. McInnes, C. Woodward, W. D. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. Peters, D. Reynolds, B. Riviere, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel,

B. Smith, X. Tang, C. Wilson, and B. Wohlmuth. Multiphysics Simulations: Challenges and Opportunities. Technical Report ANL/MCS-TM-321, Argonne National Laboratory, Dec 2011.

[2] J. R. Cary, A. Hakim, M. Miah, S. Kruger, A. Pletzer, S. Shasharina, S. Vadlamani, R. Cohen, T. Epperly, T. Rognlien, A. Pankin, R. Groebner, S. Balay, L. McInnes, and H. Zhang. FACETS – a framework for parallel coupling of fusion components. In *Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, PDP '10, pages 435–442, Washington, DC, 2010. IEEE Computer Society.

[3] N. Podhorszki, B. Ludaescher, and S. A. Klasky. Workflow automation for processing plasma fusion simulation data. In *Proceedings of the 2nd workshop on Workflows in support of large-scale science*, WORKS '07, pages 35–44, New York, 2007. ACM.

[4] W. R. Elwasif, D. E. Bernholdt, L. A. Berry, and D. B. Batchelor. Component framework for coupled integrated fusion plasma simulation. In *Proceedings of the 2007 symposium on Component and Framework Technology in High-Performance and Scientific Computing*, CompFrame '07, pages 93–100, New York, 2007. ACM.

[5] P. Chidyagwai and B. Rivière. On the solution of the coupled Navier-Stokes and Darcy equations. *Computer Methods in Applied Mechanics and Engineering*, 198(47-48):3806–3820, 2009.

[6] D. Gaston, C. Newman, G. Hansen, and D. Lebrun-Grandié. MOOSE: A parallel computational framework for coupled systems of nonlinear equations. *Nuclear Engineering and Design*, 239(10):1768–1778, 2009.

[7] T. D. Rognlien and M. E. Rensink. Edge-plasma models and characteristics for magnetic fusion energy devices. *Fusion Engineering and Design*, 60(4):497 – 514, 2002.

[8] T. D. Rognlien, X. Q. Xu, and A. C. Hindmarsh. Application of parallel implicit methods to edge-plasma numerical simulations. *J. Comput. Phys.*, 175:249–268, January 2002.

[9] John Cary (PI). Framework Application for Core-Edge Transport Simulations (FACETS). Proto-FSP project, see `http://www.facetsproject.org`.

[10] J. Cary, J. Candy, J. Cobb, R. H. Cohen, T. Epperly, D. Estep, S. Krasheninnikov, A. Malony, D. McCune, L. McInnes, A. Pankin, S. Balay, J. Carlsson, M. R. Fahey, R. Groebner, A. Hakim, S. Kruger, M. Miah, A. Pletzer, S. Shasharina, S. Vadlamani, D. Wade-Stein, T. Rognlien, A. Morris, S. Shende, G. W. Hammett, K. Indireshkumar, A. Pigarov, and H. Zhang. Concurrent, parallel, multiphysics coupling in the FACETS project. *J. Phys.: Conf. Ser.*, 180:012056, 2009.

[11] V. A. Mousseau and D. A. Knoll. New physics-based preconditioning of implicit methods for non-equilibrium radiation diffusion. *J. of Comput. Phys.*, 190(1):42–51, 2003.

[12] E. Santilli and A. Scotti. An efficient method for solving highly anisotropic elliptic equations. *Journal of Comput. Phys.*, 230(23):8342–8359, 2011.

[13] K. N. Premnath, M. J. Pattison, and S. Banerjee. Steady state convergence acceleration of the generalized lattice Boltzmann equation with forcing term through preconditioning. *Journal of Comput. Phys.*, 228(3):746–769, 2009.

[14] L. Chacón, D. A. Knoll, and J. M. Finn. An implicit, nonlinear reduced resistive MHD solver. *Journal of Comput. Phys.*, 178(1):15–36, 2002.

[15] S. P. Hirshman, R. Sanchez, and C. R. Cook. SIESTA: A scalable iterative equilibrium solver for toroidal applications. *Physics of Plasmas*, 18(6):062504, 2011.

[16] C. S. Chang, S. Ku, and H. Weitzner. Numerical study of neoclassical plasma pedestal in a tokamak geometry. *Physics of Plasmas*, 11(5):2649–2667, 2004.

[17] M. R. Dorr, R. H. Cohen, P. Colella, M. A. Dorf, J. A. F. Hittinger, and D. F. Martin. Numerical simulation of phase space advection in gyrokinetic models of fusion plasmas. In *Proceedings of the 2010 Scientific Discovery through Advanced Computing (SciDAC) Conference*, pages 42–52, July 2010, Oak Ridge, TN.

[18] R. Schneider, X. Bonnin, K. Borrass, D. P. Coster, H. Kastelewicz, D. Reiter, V. A. Rozhansky, and B. J. Braams. Plasma edge physics with B2-Eirene. *Contributions to Plasma Physics*, 46(1-2):3–191, 2006.

[19] D. Stotler, C. Karney, R. Kanzleiter, and Jaishankar S. User's guide for DEGAS 2. Technical Report Release V. 4.3, Princeton Plasma Phyiscs Laboratory, November 2009.

[20] B. J. Braams. A multi-fluid code for simulation of the edge plasma in tokamaks. Technical Report NET report Nr. 68, Next European Torus, 1987.

[21] B. J. Braams. Radiative divertor modelling for ITER and TPX. *Contributions to Plasma Physics*, 36(2-3):276–281, 1996.

[22] S. V. Patankar. *Numerical heat transfer and fluid flow*. Series in Computational Methods in Mechanics and Thermal Sciences. Taylor & Francis, 1980.

[23] X. Bonnin, D. Coster, C. S. Pitcher, R. Schneider, D. Reiter, V. Rozhansky, S. Voskoboynikov, and H. Bürbaumer. Improved modelling of detachment and neutral-dominated regimes using the SOLPS/B2-Eirene code. *Journal of Nuclear Materials*, 313-316(0):909–913, 2003.

[24] T. D. Rognlien, J. L. Milovich, M. E. Rensink, and G. D. Porter. A fully implicit, time dependent 2-D fluid code for modeling tokamak edge plasmas. *J. Nuclear Materials*, 196-198(0):347–351, 1992.

[25] Y. Saad. ILUT: A dual threshold incomplete LU factorization. *Numerical Linear Algebra with Applications*, 1(4):387–402, 1994.

[26] S. I. Braginskii. *Transport processes in a plasma*, volume 1 of *Reviews of Plasma Physics*. Consultants Bureau, 1965.

[27] P. C. Stangeby. *The plasma boundary of magnetic fusion devices*. Series on Plasma Physics. Institute of Physics Pub., 2000.

[28] M. E. Fenstermacher, S. L. Allen, N. H. Brooks, D. A. Buchenauer, T. N. Carlstrom, J. W. Cuthbertson, E. J. Doyle, T. E. Evans, P.-M. Garbet, R. W. Harvey, D. N. Hill, A. W. Hyatt, R. C. Isler, R. A. James G. Jackson, R. Jong, C. C. Klepper, C. J. Lasnier, A. W. Leonard, M. A. Mahdavi, R. Maingi, W. H. Meyer, R. A. Moyer, D. G. Nilson, T. W. Petrie, G. D. Porter, T. L. Rhodes, M. J. Schaffer, D. M. Thomas R. D. Stambaugh, S. Tugarinov, M. R. Wade, J. G. Watkins, W. P. West, D. G. Whyte, and R. D. Wood. The two-dimensional structure of radiative divertor plasmas in the DIII-D tokamak. *Physics of Plasmas*, 4:1761–1773, May 1997.

[29] R. J. LeVeque. *Finite volume methods for hyperbolic problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[30] A. Iserles. *A first course in the numerical analysis of differential equations*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2009.

[31] C. T. Kelley. *Iterative methods for linear and nonlinear equations*, volume 16 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics, 1995.

[32] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16 of *Classics in Applied Mathematics*. SIAM, 1996.

[33] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):pp. 400–408, 1982.

[34] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 1996.

[35] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.*, 193:357–397, January 2004.

[36] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Scientific and Statistical Computing*, 11(3):450–481, 1990.

[37] G. D. Byrne and A. C. Hindmarsh. PVODE, an ODE solver for parallel computers. *Int. J. High Perform. Comput. Appl.*, 13:354–365, November 1999.

[38] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Trans. Math. Softw.*, 31:363–396, September 2005.

[39] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.2, Argonne National Laboratory, September 2011.

[40] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):187–209, 1983.

[41] W. D. Gropp, L. C. McInnes, and B. F. Smith. Scalable libraries for solving systems of nonlinear equations and unconstrained minimization problems. In *Proceedings of the Scalable Parallel Libraries Conference*, pages 60–67, Mississippi State University, 1995. IEEE.

[42] B. F. Smith, P. E. Bjørstad, and W. D. Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge University Press, New York, 1996.

[43] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics, 2000.

[44] D. Hysom and A. Pothen. A scalable parallel algorithm for incomplete factor preconditioning. *SIAM J. Scientific Computing*, 22(6):2194–2215, 2001.

[45] W. Gropp, D. Keyes, L. C. McInnes, and M. D. Tidriri. Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD. *Int. J. High Perform. Comput. Appl.*, 14:102–136, May 2000.

[46] L. N. Trefethen and D. Bau III. *Numerical linear algebra*. SIAM, 1997.

[47] J. Brown, M. G. Knepley, D. A. May, L. C. McInnes, and B. F. Smith. Composable linear solvers for multiphysics. Preprint ANL/MCS-P2017-0112, Argonne National Laboratory, 2012. Submitted to the 11th International Symposium on Parallel and Distributed Computing (ISPDC 2012).

[48] B. Aagaard, S. Kientz, M. G. Knepley, S. Somala, L. Strand, and C. Williams. Pylith user manual version 1.6.1, 2011.

[49] J. Brown, B. Smith, and A. Ahmadia. Achieving textbook multigrid efficiency for hydrostatic ice sheet flow. *SIAM Journal on Scientific Computing*, 2012.

[50] S. Abhyankar, B. Smith, H. Zhang, and A. Flueck. Using PETSc to develop scalable applications for next-generation power grid. In *Proceedings of the 1st International Workshop on High Performance Computing, Networking and Analytics for the Power Grid*. ACM, 2011.

[51] L. Wang, J. Lee, M. Anitesu, A. El Azab, L. C. McInnes, T. Munson, and B. Smith. A differential variational inequality approach for the simulation of heterogeneous materials. In *Proceedings of SciDAC 2011 Conference*, 2011.

[52] Y. Shapira. *Matrix-based multigrid: theory and applications*. Numerical methods and algorithms. Kluwer Academic Publishers, 2003.

[53] R. D. Falgout and U. M. Yang. HYPRE: A library of high performance preconditioners. In *Proceedings of the International Conference on Computational Science-Part III*, ICCS '02, pages 632–641, London, UK, 2002. Springer-Verlag.